iBridge: Improving Unaligned Parallel File Access with Solid-State Drives

Kei Davis, CCS-7;
Xuechen Zhang, Georgia Institute of Technology;
Ke Liu,
Song Jiang,
Wayne State University

Parallel I/O is central to high-performance computing (HPC), and improving its efficiency is an ongoing research and development problem. In a parallel I/O system, a file is striped over multiple servers or hard disks, and requests to the file are correspondingly decomposed into a number of sub-requests that are distributed over the servers. If a request is not aligned with the striping pattern, this decomposition can produce sub-requests much smaller than the striping unit. Because disks are much less efficient in serving small requests than large ones, the net throughput of the entire system can be severely degraded by serving these smaller requests or fragments. This effect can become the Achilles' heel of a parallel I/O system that is seeking scalability with large sequential accesses.The work described in this article contributed to CCS-7 research into common runtime elements for programming models for increasingly parallel scientific applications and computing platforms [1].

We have implemented a scheme, iBridge, in the PVFS2 parallel file system that identifies requests for fragments and uses solid-state drives to serve them, thereby eliminating their detrimental effect on disk-based server performance. Our experimental results on the LANL Darwin cluster show that iBridge can significantly improve the I/O throughput for real-world HPC applications.

To meet the demand for high-throughput-data access on storage systems by highly parallel scientific and engineering applications, parallel file systems such as GPFS, Lustre, and PVFS2 have been widely adopted to manage large data files such as checkpoint/restart files and the inputs and outputs of data-intensive applications. In these file systems the files are striped over a number of data servers to take advantage of aggregate network and hard disk bandwidth, while programmers are presented with a convenient linear logical file address space.

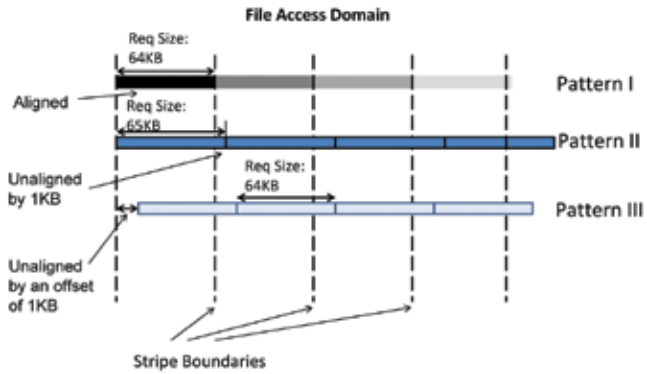

Fig. 1. File access with three different alignment patterns. In Pattern I, access is perfectly aligned. In Pattern II, sub-requests are greater than the striping pattern size. In Pattern III, the sub-requests are the same size as the striping unit but are offset, a common situation for file formats that contain header data.

With file striping, a request for a segment of logically contiguous file space is divided into sub-requests that are distributed over multiple data servers. While the striping unit size is usually reasonably large so that sub-requests are sufficiently large to obtain high disk efficiency at each server, the first and/or last sub-requests can be much smaller than the striping unit if the request pattern does not match the striping pattern–that is, data access is unaligned. Figure 1 depicts three such access patterns relative to the striping pattern.

We analyzed I/O traces from various computing environments and have discovered that unaligned access is common. One example is the set of traces of HPC applications from the Scalable I/O project at Sandia National Laboratories (SNL), including the applications ALEGRA, CTH, and S3D. As shown in Table 1, up to 62.8% (35.9% on average) of I/O requests are unaligned with the striping pattern on the data servers if we assume a 64 KB striping unit, the default for PVFS2.

Table 1. *Percentages of unaligned and random data accesses in different I/O traces with a 64 KB striping unit. Unaligned refers to requests that are larger than a striping unit (64 KB) but are not aligned to the striping unit boundaries.*

App	Unaligned (%)	Random (%)	Total (%)
ALEGRA-2744	35.2	7.3	42.5
ALEGRA-5832	35.7	6.9	42.6
CTH	24.3	30.1	54.4
S3D	62.8	5.8	68.6

These smaller sub-requests, which we call fragments, are effectively random accesses on their respective servers. Because a hard disk can be less efficient in serving random requests than sequential ones by one or more orders of magnitude, fragments and other sub-requests of the same request can be served with very different efficiency. In the case of synchronous requests the entire storage system's productivity is degraded.

To investigate the effects of unaligned access on storage system performance we ran the *mpi-io*-test benchmark in which N processes iteratively read data from a 10-GB file striped over eight data servers. Figure 2 shows that throughput for aligned access is approximately twice that for unaligned access.

For more information contact Kei Davis at kei@lanl.gov.

Funding Acknowledgments
DOE, NNSA, Advanced Simulation and Computing Program;
DOE, Office of Science

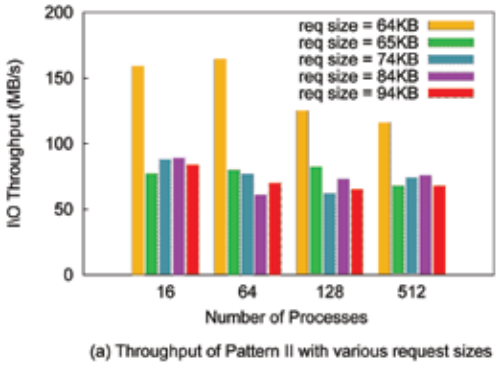

Fig. 2. I/O throughput for mpi-io-test for access pattern II from Fig. 1 for various request sizes. Note that for a request size of 64 KB access is perfectly aligned.

The objective of iBridge is to use solid-state drives (SSD) to serve fragments produced by unaligned data accesses. Such a use of SSDs represents a highly desirable combination—the SSD's strong performance advantage for random access, the fragments' relatively small sizes, and a strong need to efficiently serve fragments. However, to make the approach truly effective we had to solve a number of algorithmic problems:

- What criteria distinguish regular random requests and fragments?
- What are the criteria for deciding whether a regular random request or a fragment should be admitted into the SSD?
- Because SSD space is limited, how should it be allocated between regular random requests and fragments?

iBridge was prototyped in the PVFS2 2.8.2 parallel file system on the Darwin cluster at LANL. The cluster includes 116 48-core 2-GHz AMD Opteron nodes interconnected with a dual-rail 4X QDR Infiniband network. We configured eight nodes as data servers and one node as the meta-data server. Each data server had one 7200-RPM disk drive and a 120-GB SSD.

We sought experimental answers to the following questions, among others.

- Can unaligned requests be effectively served using iBridge in hybrid storage systems?
- Is iBridge effective for real scientific workloads with diverse data access patterns?
- How much overhead does iBridge add to the PVFS2?

iBridge was evaluated using both a range of synthetic benchmarks such as *mpi-io*-test and real-world scientific applications such as those given in Table 1, in the latter case by replaying their I/O traces rather than actual execution of the applications. Figure 3 shows the throughputs for *mpi-io*-test for both reads and writes, with and without iBridge, for 65-KB requests as a function of number of processes. In these cases, iBridge provides more than a factor of two performance improvement. Table 2 shows the service times, with and without iBridge, when replaying the HPC applications from the SNL Scalable I/O project for which alignment data was given in Table 1. With iBridge, the request service times are reduced by 13.9%, 18.7%, 25.9%, and 29.8%, respectively. Other measurements showed that iBridge added negligible time and space overhead to the PVFS2 system.

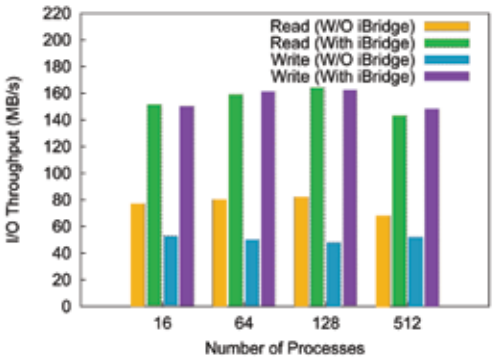

Fig. 3. Throughputs for mpi-io-test with and without iBridge as process count increases.

Parallel I/O is a cornerstone of high-performance scientific computing and continued advances in its performance are needed to keep up with advances in the rest of the computing platform. With iBridge we have shown that with a very modest investment in relatively new technology–the solid-state drive–and appropriate algorithmic development, we can significantly improve the performance of existing parallel I/O systems–and with negligible additional overhead and no changes to the application developer's interface to it.

Table 2. *Comparison of request service times when for I/O traces replayed with and without using iBridge.*

	ALEGRA. 2744	ALEGRA. 5832	CTH	S3D
PVFS2	16.6ms	17.2ms	19.4ms	36.0ms
PVFS2 + iBridge	14.2ms	14.0ms	14.4ms	25.3ms

[1] Zhang, X. et al., "iBridge: Improving Unaligned Parallel File Access with Solid-State Drives," LA-UR-12-25294; *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, (2013).